RECOGNIZING EXPRESSIVE MOVEMENT IN ARM GESTURES USING HIDDEN MARKOV MODELS

BY

MOHAMMAD MAHDI AMANZADEH

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Systems and Entrepreneurial Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Adviser:

Associate Professor Guy Garnett

ABSTRACT

Movement is one of the most basic human skills that used for communicating and interacting with the environment. Although we have an intuitive understanding of our gestures, it is hard to explain their quality. One would describe the human gestures as a collection of various actions for performing different tasks. True, but it does not explain *how* the tasks are performed, which is essential for having a more natural representation of movement. In this work we use Laban Movement Analysis (LMA), which is an analytic and experiential system for interpreting human body movement, to understand the expressive aspects of human gestures and we try to recognize them in hand movement using a supervised learning method with Hidden Markov Models (HMMs).

We first define the weight, space, and time characteristics of movement, which are described as the Basic Effort Factors (BEF) in LMA and we construct a classifier for each BEF using HMMs. We use a Microsoft Kinect to capture the body movement and try to recognize the quality of each BEF in hand gestures. Various preprocessing are done on the motion data to extract features that can describe the movement qualities. We use a windowing technique to segment the gestures into smaller sequences and allow for continuous gesture recognition. Various experiments are done to identify the optimal features and the parameters of each model, and we also address different problems in implementing the models.

Our research showed promising results in recognizing and distinguishing the BEFs of hand movement. The importance of this research is in developing systems that require deeper and more natural understanding of body movement, such as systems for recognizing musical gestures and dance, or producing computer animation.

To my parents, my uncle, and my wonderful wife Negar

ACKNOWLEDGMENTS

I would like to express my sincere thanks to my advisor Professor Guy Garnett for his valuable advices and mentoring me through this research. Also, I would like to thank my colleagues, Mike Junokas, Kyungho Lee, and Yong Hong for their suggestions and help in advancing the project.

In addition, I would like to thank Dr. Sara Hook and Kristin Carlson who helped us with performing the movement samples used in this project. I have used an abbreviation of their names, SH and KC, to address the dataset I used in each test.

This work was done as a part of the Moving Stories project which is a collaborative research in designing systems that articulate human movement knowledge with the digital technology. The research is funded by the Canadian Social Sciences and Humanities Research Council (SSHRC).

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Movement recognition is one of the most researched topics in the area of human computer interaction. Most of the previous research focuses on the recognition of specific actions in movement. Some recent work in this area tries to recognize the expressive aspects of movement to create a more natural representations of it. In this work we use Laban Movement Analysis (LMA) to describe expressive movement and we build a classifier using Hidden Markov Models to recognize some aspects of that in hand motions.

We use a Kincet device to capture the wrist joint's positions and use this data to construct the classifiers. Various preprocessing is performed on the raw data. We study the effect of each feature separately and in combination with the other motion features to identify the most effecting ones for each classifier. As we describe later, scaling of the feature values is an important part of the preprocessing and the classifiers will completely fail if we use the motion features without scaling. We use a sliding window approach to slice the gesture sequences into smaller segments to allow for real-time recognition. Finally we study the effect of different parameters of the model such as the segment size, number of hidden states and other parameters of HMM to improve the classification.

The following sections explain LMA origin and terms, and we provide a brief introduction to Hidden Markov Models.

1.1 Laban Movement Analysis

Laban Movement Analysis (LMA) is an experiential and analytical system developed by Rudolf Laban for expressing all types of the human movement.

LMA describes movement with four major components [Laban, 1980] which are Body, Effort, Space, and Shape (BESS). Body describes the structural and physical properties of the body while moving. Effort represents more subtle properties of the movement and answers *how* the body is moving. Space describes movement in terms of *where* the body moves in the environment. Shape describes the relationship and interaction of the body shape with its environments.

According to [Laban, 1980] an action has three inner stages, which are attention, intention, and decision. LMA describes the effect of inner intention on movement with four Basic Effort Factors (BEFs) that represent the qualities of movement. The BEFs are weight, space, time, and flow. Each BEF is expressed as a continuum between an indulging and fighting extreme that represent the opposite mental attitudes in performing an action.

In [Laban, 1980] BEFs are described as:

"a relaxed or forceful attitude towards weight, a pliant or lineal attitude towards space, a prolonging or shortening attitude towards time, and a liberating or withholding attitude towards flow."

Weight describes the degree of gentleness or firmness of movement and can have a quality between light (indulging) and strong (fighting). Space shows if movement is boundless and tends to move in every direction in the space or it is bounded within a specific direction, and is represented with indirect quality (indulging) or direct (fighting). Time describes how slow or fast a movement is and it is defined in a continuum between sustained (indulging) and sudden (fighting) qualities. Finally flow shows the level of fluency and liberation in movement, which can be smooth and Free (Indulging), or controlled and Bound (Fighting). The Basic Effort Factors are graphically represented with the effort continuum graph as shown in figure 1.1. The edges of graph show a continuum between the indulging and fighting states of each BEF. In Laban notation the BEFs of a gesture are shown with a subset of the edges of this graph.

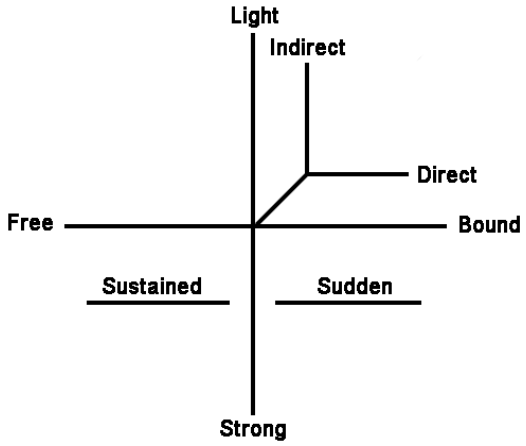

Figure 1.1: Laban Effort continuum presented in [Laban, 1980]

According to [Laban, 1980] every human action carries a combination of indulging or fighting state of weight, space, and time efforts. We can define 8 basic actions with a combination of the extreme quality of these BEFs. LMA represents these actions as the Basic Effort Actions (BEAs) which are shown in table 1.1.

BEA	Weight	Space	Time
Float	Light	Indirect	Sustained
Flick	Light	Indirect	Sudden
Glide	Light	Direct	Sustained
Dab	Light	Direct	Sudden
Wring	Strong	Indirect	Sustained
Slash	Strong	Indirect	Sudden
Press	Strong	Direct	Sustained
Punch	Strong	Direct	Sudden

Table 1.1: Basic Effort Actions

When performing an action, the amount of emphasis we put on the BEFs shows various types of that action. For example pushing an object is a type of press BEA (table 1.1) and the amount of weight we put on pushing depends on the weight of the object we push, which can be a piano that needs a gesture with great emphasis on having strong weight, or it can be a coffee table which is still heavy but the gesture has less emphasis on being strong.

If we change the strong weight of a press BEA to light, we have a glide BEA that still looks similar to a press but with lighter weight. In the example of pushing a piano, if we exchange the piano with an iron, the action of pushing the iron, or ironing, is a glide BEA.

In this work we try to distinguish the indulging or fighting state of BEFs in various examples of the eight Basic Effort Actions. Each BEA is performed with equal emphasis over the BEF qualities and they are all along the same path starting from the center of body towards the upper corners.

1.2 Hidden Markov Models

Hidden Markov Models (HMMs) have been widely used in modeling sequential data or signals. The architecture of a basic HMM assumes that a chain of hidden states with Markov property controls the properties of an observed signal. Markov property states that the probability of an observation depends only on the hidden state of model and the probability distribution of the future states only depends on the current state of the system. This property allows HMMs to efficiently model the sequential data. [Rabiner, 1989] provides a comprehensive introduction of HMMs and different examples of using them in speech and connected words recognition. This section is mostly a summary of important parts in [Rabiner, 1989].

Figure 1.2 shows the general architecture of a Hidden Markov Model. A basic HMM assumes that the probability of observing an output $\mathbf{o}_t$ in the observed sequence ($O = \{\mathbf{o}_1, \ldots, \mathbf{o}_T\}$) depends only on the hidden state of model at time t. Each observation $\mathbf{o} = [o_1, \ldots, o_k]$ is a vector of K outputs that we later refer them as the motion features. The hidden states carry different probability distributions of the outputs. The set $S\{s_1, s_2, \ldots, s_N\}$ represents N hidden states of the model and the sequence of actually observed states are denoted with $Q = \{q_1, q_2, \ldots, q_T\}$. As we described earlier, an HMM assumes that signals have Markov property. In basic HMMs this property has the order one, which means that the probability of having s_j at time t only depends on the probability of the actually observed state at time

$t-1$ (q_{t-1}).

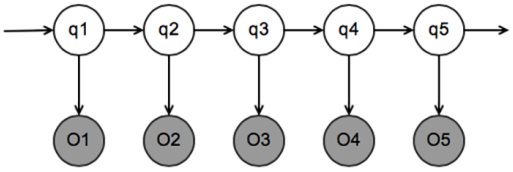

Figure 1.2: The general architecture of a Hidden Markov Model

We formally define an HMM with $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ and it represents the probability values of the edges of HMM graph. The $N \times N$ matrix $\mathbf{A}$ is the state transition probability and each element of it ($a_{ij} = P[q_{t+1} = s_j | q_t = s_i]$) contains the transition probability between states i and j. The $K \times N$ matrix $\mathbf{B}$ represents the observation probability, where the column $\{\mathbf{b}_i = P[\mathbf{o}_t | q_t = s_j]\}$ contains the probability of observing the features when the system is at state s_i. Finally the vector $\pi = [\pi_1, \ldots, \pi_N]$ represents the probability that we have state i at time $t = 0$ ($P[q_0 = s_i]$).

In an HMM that has first order Markov property, the state transition probability A is defined with which means that the probability of observing a state at time $t+1$ only depends on the observed state at the current time. The probability of observing an output only depends on the current state of system and is defined with $b_j(k) = P[O_t | q_t = s_j]$.

Three main problems of modeling signals with HMMs are addressed in [Rabiner, 1989], which are:

1. How to efficiently compute the probability of an observation, or $P(O|\lambda)$
2. What is the optimal state sequence for an observation
3. How to compute the model parameters that maximize the probability of an observation $P(O|\lambda)$

To answer the first problem we assume that we have a trained model and we would like to identify the likelihood of a query sequence given the model. To compute $P(O|\lambda)$ we need to find the probability of every possible

state sequence for an observation which will lead to equation 1.1. The basic solution of this problem needs $2T \cdot N^T$ computations because at each layer we need to compute all the combination of the state at current time with the state at the next time. To solve it efficiently, we can use the forward step of the forward-backward algorithm. This algorithm is a dynamic programming method that breaks down this complex problem into smaller parts. It computes the information passed between the states in consecutive time using the information that have reached from the previous states as you can see in figure 1.3. The forward algorithm starts this computation from time $t = 1$ and gradually computes the observation likelihood at each time. This process reduces the original computation time to N^2T.

$$P(O, Q|\lambda) = P(O|Q, \lambda)P(Q, \lambda) \tag{1.1}$$

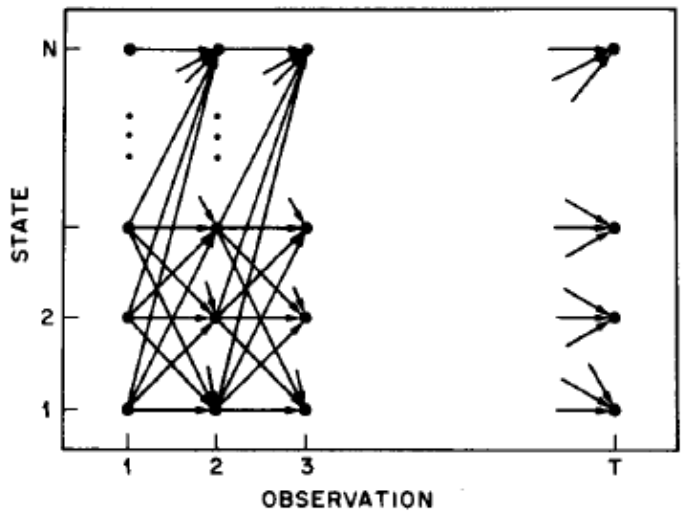

Figure 1.3: The graph shows different possible paths for computing the $P(O, Q|\lambda)$. Image adopted from [Rabiner, 1989]

Computing the optimal state sequence for an observation depends on how we define the optimal. An optimal sequence can be a sequence with states that are individually the most probable state at each time, regardless of their previous states, or it can be the most likely pairs of them (q_{t-1}, q_t) or triples and higher orders. The Viterbi algorithm is an algorithm that uses the forward-backward algorithm to solve this problem. It uses the forward step

and computes the probabilities starting from time $t = 1$. The algorithm uses the state that maximizes $P(S_t|q_{t-1})$ to compute the probability at time $t + 1$ ($P[S_{t+1}|q_t]$). After identifying the last optimal state (q_T) it uses the backward step that repeats the same procedure in opposite direction to find a sequence of the states that will end with (q_T) and maximizes the observation probability.

According to [Rabiner, 1989] there is no optimal way to adjust the model parameters and maximize $P(O|\lambda)$. One possible way to maximize this likelihood is using Expectation Maximization (EM) which is a method for estimating the parameters of statistical models. This method starts with an initial guess for the model parameters and in the E step it computes the likelihood of model using the training data. In the M step the algorithm tries to find new parameters that maximize the likelihood and iteratively replacing them with old parameters and repeating this procedure. Baum-Welch algorithm is an EM method used for HMMs and it uses forward-backward procedure to compute the expectation of the parameters. Since this algorithm will potentially result only in a local maxima, we have to train the HMM multiple times to ensure that we find the best parameters for the model.

Finally to compute the probability of an observation we have to consider if the observation is discrete or continuous. The general architecture of HMMs assumes that the observations are discrete and are randomly generated from a set of finite symbols. To model continuous signals we have to put some restrictions on the observation probability distribution and define them with a probability density function that allows the observation probabilities to be estimated continuously. According to [Ghahramani, 1998] depending on the type of the signal we can model continuous observations with a single Gaussian, mixture of Gaussians, or neural networks. [Rabiner, 1989] describes modeling with mixture of Gaussians, and also it discusses the auto-regressive HMMs that are used to model the continuous observation with Gaussian autoregressive features.

1.2.1 Varieties of Hidden Markov Models

The model that we discussed so far is a general architecture of HMM that has a single layer of hidden states with ergodic property, where each state can be reached from every other states. There are various types of HMM that alter the assumptions in basic HMM to make appropriate model for different problems. In this section we briefly discuss some of these models.

[Rabiner, 1989] describes different architectures for the hidden states of HMM that has a single layer of state space. One of these architectures that has been explored in this work is the Left-Right HMM. This model modifies the ergodicity of the state transition graph and assumes that a state can only be reached from the states that have a lower rank. This can be shown using an upper triangular state transition matrix, which only allows a state to jump to the states that have a higher rank. We can put an additional constraint on the transitions and limit them to a maximum size of jump. According to [Rabiner, 1989] this model was successful in speech processing.

A basic Hidden Markov Model assumes that the system is a Markov process with a single layer of discrete hidden states that control the probability distribution of observations. According to [Brand et al., 1997] this restrictive assumption prevents the model to have good performance in modeling the problems that naturally have more than one hidden control layer such as in vision and speech. With Dynamic Bayesian Networks (DBN) we can generalize the concept behind HMM by describing the flow of information between the hidden and observed nodes. As a result we can generate variants of HMM with multiple hidden layers using DBNs. [Murphy, 2002] provides a comprehensive introduction on DBNs and provides examples of HMMs that have multiple layers of states.

[Ghahramani, 1998] introduces HMMs that have multiple layers of hidden states. Using multiple hidden layers can help in capturing a deeper underlying structure of movement rather than modeling the hidden states with Gaussian models. Factorial HMMs is an example of these models. This model assumes that the observation depends on layers of hidden states each with a distinct distributions. Tree Structured HMM is a variety of the factorial models that

has state layers with dependent distributions. Switching State Hidden Markov Models (SHMMs) are another type of HMMs that has a layer of hidden state that can have multiple continuous features and a state sequence with discrete values that switches between different distributions of the continues features and they both control the probability of an observation.

Hierarchical HMM (HHMM) is another extension of HMMs that is used in modeling the sequences with nested structure such as in language where the sentences are constructed with different words and each word is made with phonetic units. As introduced in [Fine et al., 1998], an HHMM has two types of hidden states, which are the states that can generate an observation, and the internal states that generate a sequence of states. We can use this model in gesture recognition by representing different gestures with the internal states and using the observation states to model each gesture.

Coupled Hidden Markov Model (CHMM) introduced in [Brand, 1997] for modeling multiple process signals. The proposed model showed less sensitivity to the initialization conditions comparing to the HMM. This model assumes that a process is generated from two independent sub processes that do not have a direct effect on each other. The CHMM is made from a combination two HMMs, where its state sequence is the Cartesian product of the states of the two HMMs. An example of using this CHMM could be in modeling the gestures of two tennis players, where their action at each time depends on their own action in the previous time and also the opponent's action in the previous time.

1.3 Thesis Outline

In this work we addressed various problems in designing a system to recognize the weight, space, and time of hand movement when performing various samples of the eight Basic Effort Actions using Hidden Markov Models. In the next chapter we discuss some related research in the area of gesture recognition. Chapter 3 covers the motion capture and process of identifying salient movement features for each Basic Effort Factor. In chapter 4 we discuss

the problems related to modeling the BEFs with our HMM classifier. In this chapter we analyze the effect of scaling the feature values on the performance of the HMM and we describe the sliding window technique used for real-time gesture recognition. Finally in chapter 5 we explore the different issues of our classifier and performed various tests to improve its performance.

In most of the experiments done in this research we use uniform parameters to be able to compare the models side by side. The parameters are identified with experiment and we use the setting that worked best in most of the examples. Unless otherwise is stated:

- The HMMs have 8 hidden states and each state is modeled with a mixture of 2 Gaussians
- The Gaussians have diagonal covariance, and are initialized with k-means algorithm. All the other parameters of model, such as initial state probabilities, state transitions, and the initial probability of Gaussian mixtures are initialized randomly
- We scale the features using the inverse variance of each feature computed from the training data, and we use a sliding window with size 8 frames
- We use a moving average filter of size 8 on the motion velocity and use that to train the HMMs.

CHAPTER 2

RELATED WORK

Human gesture recognition is approached with different methods in the reviewed literature. Some of them focus on the techniques and algorithms used for gesture recognition. A group of these works use variants of Dynamic Bayesian Network such as HMMs or Conditional Random Fields (CRFs) as we can see in [Lee and Kim, 1999] and [Chi et al., 2000]. [Vasilescu, 2002] used Singular Value Decomposition to decompose a motion corpus of various actions performed by different people into a triple of the individual's motion signature, type of motion, and different actions. [Black and Jepson, 1998] proposed a model for gesture recognition using the condensation algorithm. This method can be seen as a combination of HMM and dynamic time warping and tries to model a gesture with s a combination of template trajectories in different scales.

In order to perform gesture recognition in real-time, we need to have some information about the beginning and end of gestures. [Lee and Kim, 1999] proposed a threshold model for identifying the non gesture sequences in a continuous movement and recognize the beginning and end of the actual gestures using the threshold model. In this method they train a left-right HMM for each gesture and also a threshold HMM using the learned hidden states of all the other models. the threshold model always outputs a high likelihood to input motion until one of the gesture HMMs generate a higher likelihood. They use this point to identify the beginning of a gesture. [Bouchard and Badler, 2007] proposed a method for segmenting motion by identifying the places where we have a change in effort factors of movement using neural networks. [Zhao and Badler, 2005] segmented the movement by tracking the zero crossing of acceleration and motion curvature features. Their experiments showed that the motion curvature has a high value when the movement starts and stops and also when we have a change in its direction.

A group of the reviewed literature work on modeling the conversational gestures. [Zhang et al., 2004] used a two layer HMM for recognizing different human actions in a meeting. The model recognizes the actions of individuals using the HMM in first layer, and uses the result of this step as an input for the HMM in the second layer to recognize a group action. [Levine et al., 2010] uses HMM to recognize a hidden distribution on the conversational gestures and uses them as the hidden states of a CRF to model the prosody features of that conversation. Later they use a reinforcement learning method to select gesture segments that are related to a conversation to animate the characters that are engaged in a spoken conversation.

Some other research in this area try to model deeper characteristics of movement and answer *how* the gestures are performed. Most of these works use the effort and shape category of LMA to describe expressive movement and identify the motion features that correlate with them. Various techniques are used to model the effort factors, such as neural networks in [Zhao and Badler, 2005] or hierarchical HMM in [Alaoui et al., 2014]. [Chi et al., 2000] used LMA to generate natural animations by imposing the parameters that control shape and effort of movement in characters. They use the shape parameters to modify the position of character at each key frame, and they control the expressive aspects of character's movement using the effort parameters.

Identifying features that can describe the effort factors is a challenging part in the related research. Majority of them used a combination of motion derivatives and curvature features to identify the space and time efforts. Weight is the most problematic effort to model in these works. [Alaoui et al., 2014] believes that weight can be described with contraction and release of of the muscles and used EMGs to capture the muscle tensions. [Zhao and Badler, 2005] also used a motion sensor near the sternum area to capture the muscular tension.

CHAPTER 3

MOTION FEATURES

One of the important steps in modeling the Basic Effort Factors (BEFs) of movement is identifying the motion features that can describe the BEFs regardless of the movement trajectory. We can perform a single movement with different expressions. For example imagine the gesture that you pose when saying "but why?". The way that this gesture is expressed in a normal situation can be totally different from posing the gesture with anger. Therefore we need to find features that can describe the quality of performing a gesture.

The focus of this research is on recognizing the expressive gestures by tracking the movement of wrist joint. We record various samples of the eight Basic Effort Actions that are performed with the right or left hand. Each gesture puts the maximum emphasis on either the indulging or fighting quality of each BEF. We extract various motion derivatives and curvature features from the positional data and analyze their correlation with each BEF.

3.1 Motion Capture

3D Motion Capture (mocap) systems are widely used for collecting the movement data for gesture recognition. These systems can track the position of various body parts using the techniques in computer vision and some of them use markers that are attached to performer to have more accurate data. Although the marker-based systems are more accurate, the recording session require more constraints and the cost of data collection is much higher than a marker-free system. In this work we used Microsoft Kinect V2, which is a marker-free mocap system that can capture the 3D position of various body

joints at 30 frames per second. Kinect has an infrared camera to understand the depth, and it uses computer vision techniques on the video and infrared inputs to track the position of body joints.

We asked two individuals trained in LMA[1] to assist us in recording eight Basic Effort Actions (BEA). The recording sessions are done separately to ensure that the performers can freely express the gestures. Each gesture was performed on a similar path with hands starting from the center of the body towards the upper right or left corner and retracting to the beginning position. We recorded the gestures in three steps. At each step we asked the performer to put maximum emphasis on expressing one of the three BEAs and recorded six samples of each BEA using the right and left hand. Since the horizontal axis of the wrist positions is inverted in right and left hand gestures, we multiplied the horizontal axis on the left hand data with a -1 to align them with the positions of right hand.

For each dataset we collected 18 samples of the BEAs. At first, we removed the non-gesture frames prior to the beginning and after end of actual movement. With experiment we realized that the norm of velocity (speed) of these parts is lower than 0.005 centimeter per frame. Table 3.1 shows the average speed and length of gestures with sudden and sustained Time in our datasets. As we described in the introduction sudden gestures are normally faster than the sustained gesture. You can see in the table that the average speed of sudden gestures is more than twice of the sustained gestures and they have less than half of the length of the sustained gestures.

	sudden		sustained	
	average speed	average length	average speed	average length
SH dataset	0.0364	35	0.0134	91
KC dataset	0.0268	40	0.0130	98

Table 3.1: Average speed and length of the gestures in each dataset

[1]We use abbreviations *SH* and *KC* to address each dataset

3.2 Feature Extraction

The raw motion capture data cannot be directly used for identifying the Basic Effort Factors (BEFs). Our hypothesis is that the positional data will limit the system to the exact movement trajectories in 3D space. We use three main groups of features to explore a measurable information about each BEF. We extract the derivative of positional data with various orders to obtain information about the changes in position, velocity, and acceleration of the gestures. We also use different curvature features such as dot or cross product in various intervals. The third group of features is the Fast Fourier Transform (FFT) of the previous features that contains information about the frequency components of the primitive features.

Velocity, acceleration, and jerk of motion are used in the derivative feature group. The derivative of positional data, or velocity, reveals the changes in the joint position at each frame. The second derivative of position is acceleration which shows the changes in the velocity of movement, and the third order derivative is jerk. These features are formally defined with $\mathbf{v}$, $\mathbf{a}$, and $\mathbf{j}$ vectors in equation 3.1. We denote the $\{x, y, z\}$ position at each time with $\mathbf{p}_t$.

$$\begin{aligned} \mathbf{v}_t &= \mathbf{p}_{t+1} - \mathbf{p}_t \\ \mathbf{a}_t &= \mathbf{v}_{t+1} - \mathbf{v}_t \\ \mathbf{j}_t &= \mathbf{a}_{t+1} - \mathbf{a}_t \end{aligned} \tag{3.1}$$

We also compute the average value of these features in different time scales using a moving average filter to better understand the time scale over which the efforts are significant. According to [Smith et al., 1997] moving average is a reasonable smoothing filter for reducing the random noise while keeping the major changes in the signal. Using larger window size for this filter will result in a smoother output. We define the moving average filter using window size w in equation 3.2. Also in Figure 3.1 we show the result of applying this filter with various window sizes on the norm of velocity.

$$\bar{x}_t = \frac{1}{w} \sum_{\tau=w-\frac{t}{2}}^{w+\frac{t}{2}} x_t \tag{3.2}$$

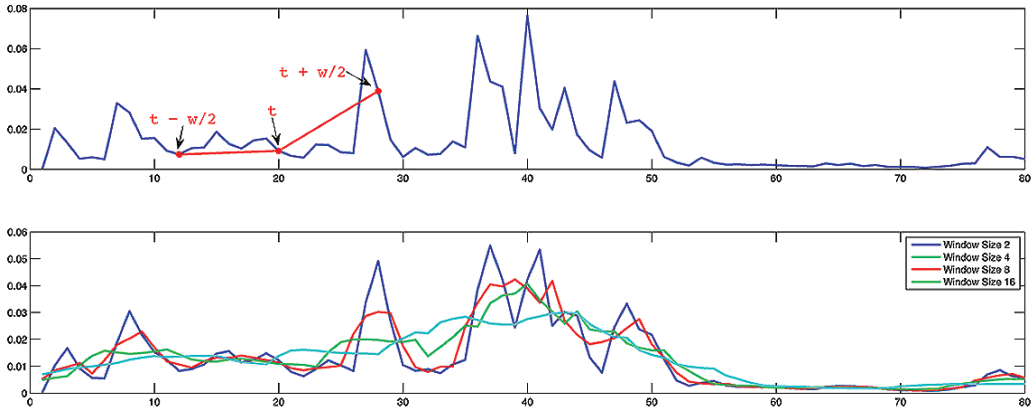

Figure 3.1: Effect of the moving average filter on the norm of velocity

Different Curvature features are used to measure the deviation of movement trajectory from a straight line. Examples of them are dot product and cross product of the change in joint position in consecutive intervals. These features are sensitive to the changes in movement direction and can be computed in various intervals to assess the trajectory curve in longer frames. Figure A.1 shows the dot product of wrist position using an interval of size w. We can see the dot product of this gesture with different window sizes in figure 3.2. Our observation showed that using larger window sizes for the curvature features makes them more sensitive to subtle changes in curvature. Complete description of the curvature features we used in this work can be found in appendix A.

In order to analyze the derivative and curvature features in frequency domain, we compute their Fourier transform at each frame and use the magnitude of the Fourier spectrogram to train the classifiers. Here the features are the frequency components of primitive features that are extracted through the Fourier transform. To compute the Fourier transform at each frame we use the sliding window technique that we used in moving average filter, but this time instead of taking the average of feature values inside the window, we extract its Fourier transform. By increasing the window size we can increase the frequency resolution which means more intermediate frequency components between 0 and 15 Hz[2] (figure 3.3). Although this action will result in decreasing the time resolution, due to the well known time-frequency tradeoff.

[2]Sampling rate of Kinect is 30 fps, therefore the maximum resolvable frequency (Nyquist) is half of the sampling rate, that is 15 Hz

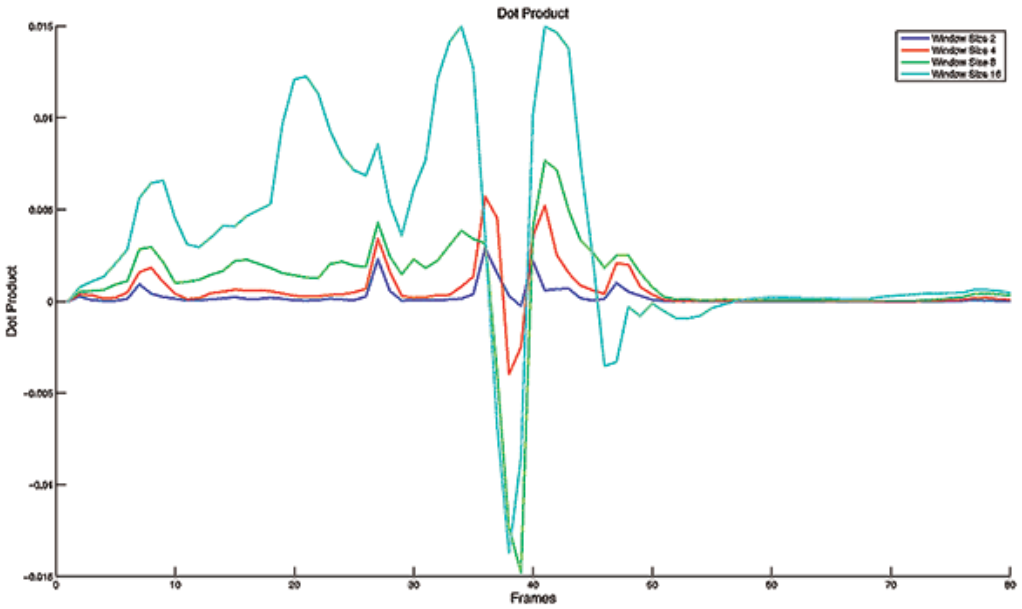

Figure 3.2: Dot product of a sample gesture with different sizes

The Fourier transform is computed over a period of the signal. Therefore this feature contains information about the entire signal during that period. Since we want to use it as a feature per each frame, it needs to put more weight on the signal values that are closer to the center of the window. To help with this problem we can apply a Hann window function on the signal, and put more weight on the central frames. Figure 3.4 shows the effect of using a Hann window function on the spectrogram of a velocity feature. As you can see in the spectrogram with uniform window function, the areas with higher energy are not sync with the actual signal, while the Hann window function can fix this problem.

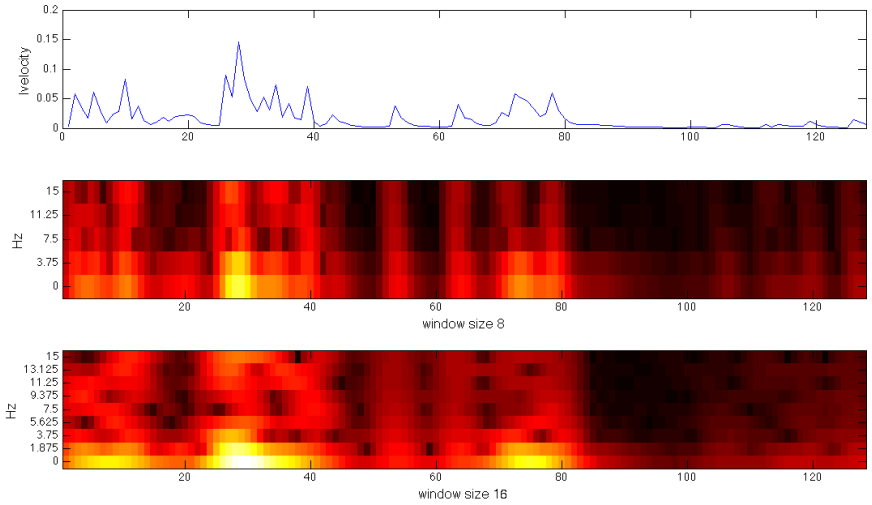

Figure 3.3: Comparing the frequency resolution of the Fourier spectrogram using window size 8 (top) and window size 16 (bottom)

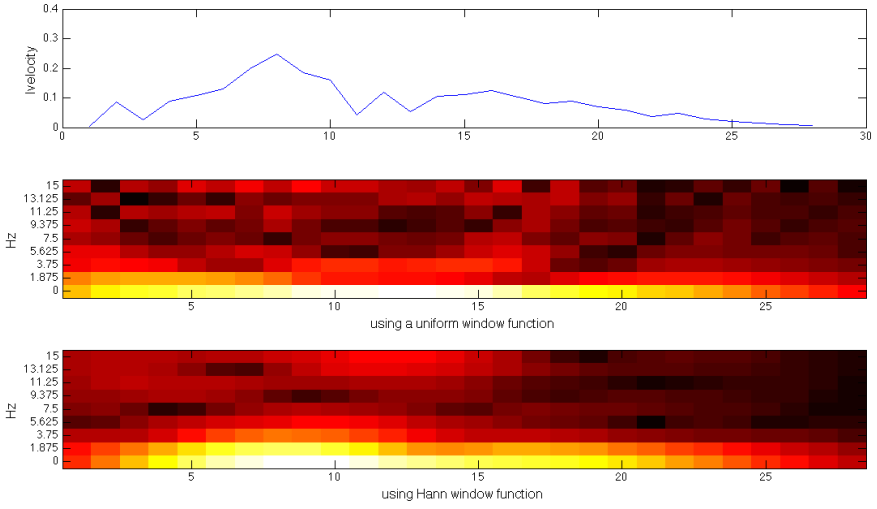

Figure 3.4: Comparison of using rectangular and Hann window function in computing the Fourier transform

3.3 Identifying the Salient Features

Different combination of the proposed features are used to identify the weight, space, and time of the recorded gestures in SH dataset. A Hidden Markov Model (HMM) is trained for each indulging and fighting quality of the BEFs and we compare the likelihood of the models to identify the most likely class. Each movement sequence is segmented into 16 frame parts with 15 frames overlap to be able to perform the evaluation in real-time. Complete description of the system design can be found in chapter 4. Figure 3.5 shows the comparison of the ROC curves of the derivative and curvature features, and figure 3.6 shows the comparison of Fourier transform of these features.

As we described earlier, the time effort describes how fast or slow is a movement. Our initial hypothesis was that the velocity could be a good measure for this effort. Experiments in[Rett et al., 2010] showed that the velocity and acceleration have high values in gestures with sudden time and [Alaoui et al., 2014] used the norm of jerk for the time feature.

In our experiments as you can see in figure 3.5, using a moving average filter with window size 8 on a combination of velocity, acceleration, and jerk resulted in 87 percent accuracy in recognizing the time effort. We could also get roughly the same result by using a combination of velocity feature and cross product with 8 frame intervals. These features resulted in about 10 percent more accuracy comparing to the velocity and 20 percent comparing to the acceleration.

The space effort shows if the movement is bounded or it tends to go in every direction. With this definition we first guessed that curvature features might be good options for describing the space effort. [Zhao and Badler, 2005] and [Alaoui et al., 2014] used a measure of distance between elbow and the chest as the space feature. [Zhao and Badler, 2005] also used the rate of change in the tangent vector of the trajectory which basically the velocity feature we used in this work. Their experiments showed that this feature has high values along the trajectory of indirect movement and also at the points near the direction change in direct gestures.

As you can see in figure 3.5 our experiments confirmed that the velocity is a good feature for the space effort. We could get slightly better accuracy using a moving average of size 8 on a combination of velocity, acceleration, and jerk however this needs three times more computation time. Our results showed about 80 percent accuracy in recognizing the space effort. The feature comparison showed that non of the proposed curvature features work as well as these two features, except the cross product with 8 frames interval and a combination of velocity feature and dot product with 8 frames interval.

Identifying salient features for the weight effort is the most challenging part in the related research. As we described earlier this effort describes the degree of gentleness or firmness of movement which is hard to capture using a video based motion capture device. [Chi et al., 2000] states that gestures with strong weight then to have higher acceleration. However they use this feature to enforce weight on animated characters not for effort recognition. According to [Alaoui et al., 2014] the weight effort can be described with contraction and release of of the muscles and used EMG sensors to capture this feature. [Zhao and Badler, 2005] also believes that weight can be captured with muscle tensions but they used a sensor over the sternum area and captured the excursions in vertical movement of sensor.

Our initial guess was that we might be able to capture the muscle tensions from the Fourier transform of velocity feature because the tension in gestures with strong weight might result in greater values in high frequencies comparing to the gestures with light weigh. However as you can see in figure 3.6, non of the Fourier transformed features worked as well as the primitive features. As you can see in figure 3.5, using an 8 frame moving average filter on the velocity resulted in the highest accuracy in weight recognition.

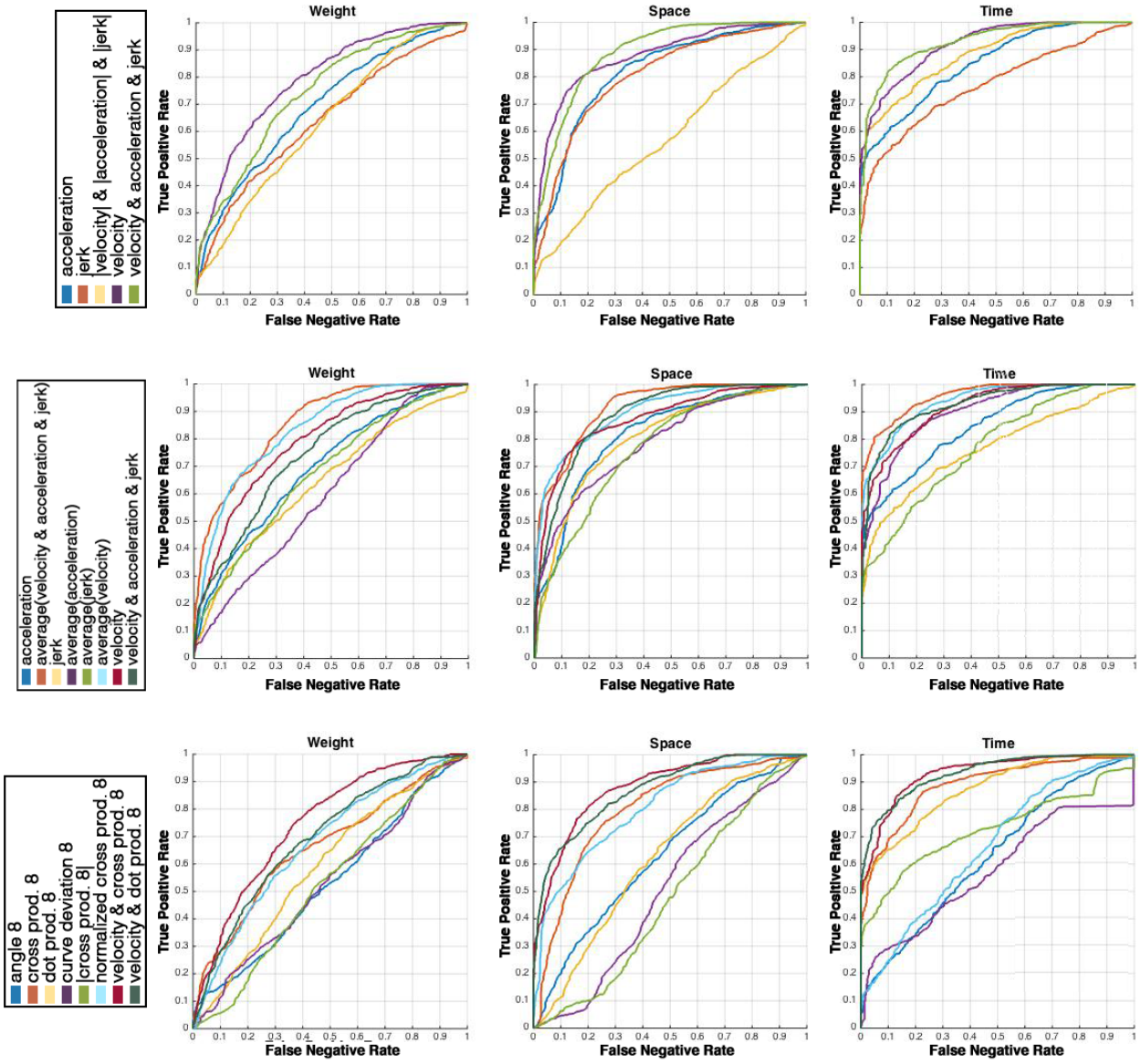

Figure 3.5: Comparison of the curvature and derivative features - positive class is the indulging effort, negative class is the fighting effort

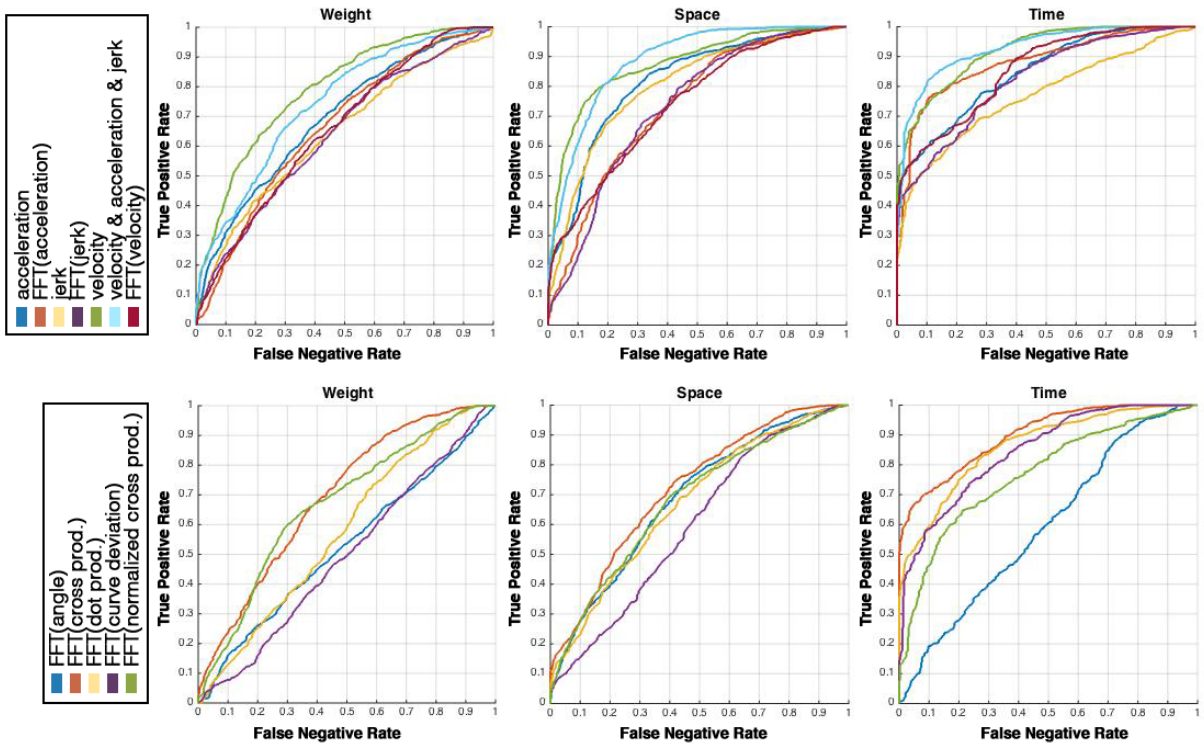

Figure 3.6: Comparison of the Fourier transform of primary features - positive class is the indulging effort, negative class is the fighting effort

CHAPTER 4

SYSTEM DESIGN

To recognize the Basic Effort Factors of movement we designed a classifier for each weight, space, and time effort. Each indulging and fighting quality of the BEF is modeled with a Hidden Markov Model. Figure 4.1 shows the diagram of a weight classifier. The classifier predicts the effort quality of an input sequence by comparing the likelihood of the two HMMs and chooses the model that produces higher likelihood.

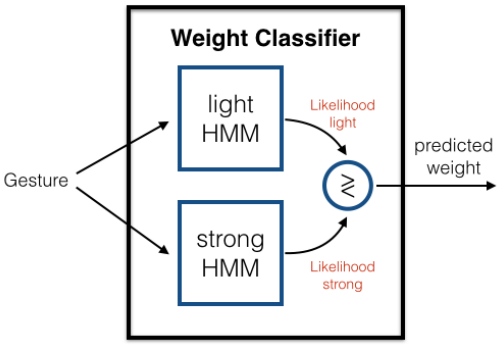

Figure 4.1: Classifying weight with HMM

According to [Brand et al., 1997] HMMs are prone to result in models with different qualities. This is because the HMMs are trained using an Expectation Maximization method as we discussed in the introduction. The EM algorithms does not guarantee to converge to a global solution and they are sensitive to the initial values of the model parameter [Bailey et al., 1994]. Since in this work we initialize the HMM parameters with random values, we need to repeat the training multiple times with different starting values to ensure that the trained parameters are close to a global solution. Therefore we train each HMM 10 times with random starting points and chose the model that results in highest training likelihood to ensure we have a good model. Figure 4.2 shows the likelihoods of 10 different training of an HMM.

As you can see three of the trainings do not converge to good parameters.

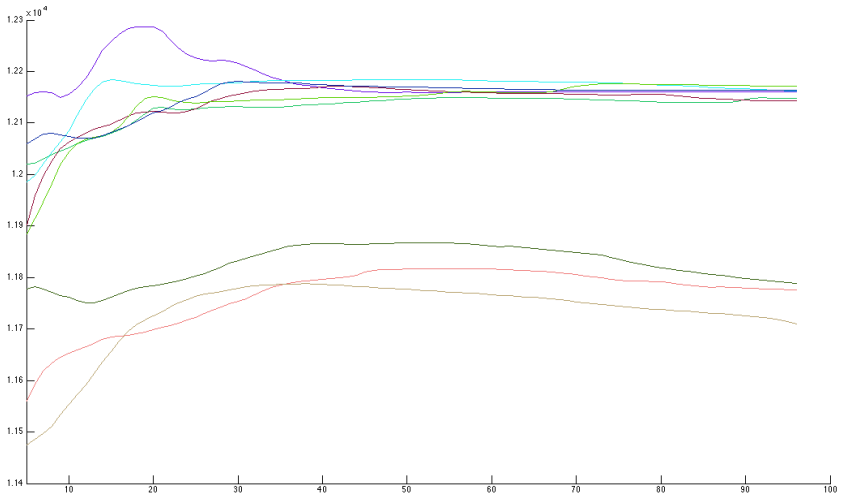

Figure 4.2: Likelihood of 10 repeated trainings of an HMM

After constructing the feature matrix we need to pre-process the input data. The first step is scaling the features which is described in the following section. Then since we would like to run the classification in real-time we need to slice the sequences into smaller segments. This is because an HMM receives an entire sequence and outputs a likelihood measure given that sequence. Therefore in order to have real-time classification we need to continuously classify smaller portions of the movement. The segmentation process is described in section 4.2.

4.1 Scaling

Our preliminary experiments showed very poor performance. To analyze this problem we examined the model parameters and the distribution of data in the states. Figure **??** shows the state distribution of an HMM trained on gestures with indirect space. As you can see in state 3, the data have zero mean and very low variance. This is while the other states have larger variance and wider distribution. This causes the probability density function of this state to have large value around zero. Comparing to the other states,

this density is very large and will cause the other states to have a small impact on the overall likelihood.

A reason for the small variance is because a large portion of this data are close to zero, as you can see in the first sub plot of figure **??**. The plot shows the velocity of indirect gestures in Z axis. We observed the same problem with the velocity of the other axes. The direct gestures also showed the same problem. This will result in a large likelihood of the sequence in both HMMs. Since we classify a sequence by comparing the likelihood of the two HMMs, this will result in a bad classification.

One way to go around this problem is to scale up the feature to have larger variance. In order to see how the scaling affects the recognition score, we used a range of numbers between 2 and 100 to scale up the velocity and compared the F1 score of the recognition. As you can see in figure 4.3 the F1 scores improved by increasing the scaling factor.

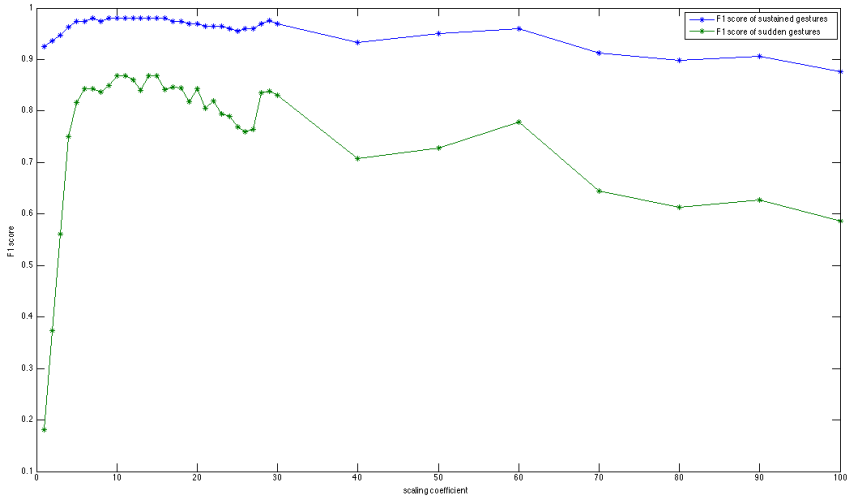

Figure 4.3: F1 score of recognizing Time by increasing the scaling coefficient

We can scale up the features by their inverse standard deviation to force them to have unit standard deviation. Another way of scaling is to use their inverse variance. By using either of these scaling parameters we could improve the performance of classifier. However using variance resulted in

slightly better results. The reason that variance is a better scaling factor is because in computing the variance we put more weight on the data points that are further from the mean comparing to the closer points, but the standard deviation is the square root of variance and shows the average deviation from the mean in distance units. As we can see in figure **??** most of the data have values around zero and this will cause the standard deviation to ignore the effect of data points that are further than zero. Therefore variance can show the spread of data better and is a better measure for scaling.

4.2 Real-Time Movement Recognition

In order to have an optimal recognition in real-time we need to predict the effort qualities for each frame of the input sequence in less than the capturing frame rate which is $\frac{1}{30}$ of a second for the kinect device. The HMM can output a likelihood score given a sequence of the input data. Therefore we cannot use only one frame for classification. Also we cannot use a long sequence because the length of a gesture could be any number of frames, and if this length is shorter than the segment size the classifier might fail to recognize the correct label. Therefore we need some way to segment the input data stream into smaller chunks and perform the classification at each frame.

One way for segmenting the input sequence is to find the points where a gesture is changing. An example of this is [Lee and Kim, 1999] where the authors train a separate HMM for non-gesture movements using the state parameters of all of the gesture HMMs. This model outputs a high likelihood all the time until one of the other HMMs picks a gesture and produces a higher likelihood. They use this point to identify the beginning and end of the picked gesture. A problem of this model is that it can have a poor performance if we don't have robust models for the gesture HMMs. Another method for segmentation is by segmenting the movement sequence into meaningful parts. According to [Zhao and Badler, 2005] one way to segment a movement sequence into meaningful parts is by tracking the zero-crossing of the second derivative of motion (acceleration) and the value of the curvature feature. They state that curvature feature has high value when the direction of motion

changes or when it stops or starts from a still position.

A different method for segmenting a sequence is using a sliding window. In this method we extract a fixed length of the input sequence at each frame backwards in time, and assign the likelihood of this sequence to the last frame of the window. [Bevilacqua et al., 2010] used the sliding window technique for classifying gestures with a Left-Right HMM in real-time. They use windowing to reduce the number of frames needed for computing the likelihood of a sequence and claim that their test results show this method was successful.

We use the sliding window technique for real-time classification. In order to find the likelihood of a sequence we just need to compute the forward parameter at each frame of the sequence. As we described in the introduction chapter, this method only needs N^2T computations which is small number if we have a short window size and a few hidden states.

For training the model we slice each training sequence into segments with size equal to the window size and no overlap. The reason for not using the entire sequence for training is because we want the model to optimize its parameters using only the information about the frames inside the window. We do not use overlapped sequences for training to avoid redundant sequences in training. When a window is greater than the remaining of a sequence or the entire sequence, we just use the remaining frames in the segment. For testing, we use the same window size, but this time the window moves one frame at each frame and we assign the predicted effort quality of the segment to the ending frame.

Figure 4.4 compares the F1 scores of identifying BEFs using various window sizes. The windowed gestures are also compared with a model that uses entire gestures for training and testing. As you can see using the entire sequence will result in better classification. However in practice since we do not know the beginning and end of the gestures and also we want to classify in real-time, we cannot classify the gestures using their entire sequence. You can see that by increasing the window size the result of classification gets closer to the baseline model. However using larger window size requires more delay in prediction. A window size of 16 means that we will have at least 0.5 second

delay because the frame rate of kinect is 30 fps. Therefore the window size depends on the amount of delay that we can expect from the system. Using window size 16 we have a reasonable trade-off between delay and the accuracy.

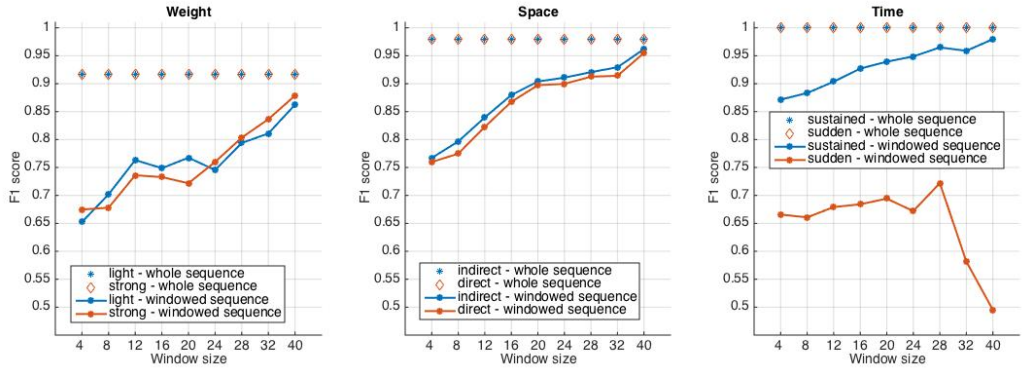

Figure 4.4: Comparison of the F1 scores using different window sizes for segmentation

CHAPTER 5

EVALUATION

In this chapter we perform various analysis on the models and motion data. In section 5.1 we modify different parameters of Hidden Markov Model and compare their performance. Different number of states and Gaussians are compared for each classifier. We also compare the performance of classifiers using HMMs with fully connected states (ergodic) and the left-right architecture. In section 5.2 we address different issues that we encountered in modeling the BEFs and try to improve the performance of classifiers. Finally in section 5.3 we evaluate the classifier using two different datasets.

5.1 Tuning HMM Parameters

We modified various properties of the Hidden Markov Model and compared their performance. Different number of states and Gaussians are used to model the BEFs with HMMs. We also compared The following describes our experiments in identifying the parameters that worked best for the HMMs and also we compared the performance of an the HMMs with fully connected state space with left-right models.

An important step in utilizing the maximum capability of the HMM is identifying the best number of hidden states to model each class. We also need to identify the right number of Gaussian mixtures for modeling the data in each hidden states. [Siddiqi et al., 2007] proposed a method for identifying the number of hidden states that best describe the data. In this method the HMM is trained with an algorithm that iteratively splits the states into the parts with similar dynamics. Although this is a good method for finding the right number of states, it needs to modify the learning process of HMM.

In this work we use a manual approach to identify the right number of hidden states and Gaussian mixtures to model the BEFs. For this reason we train the HMMs with various number of hidden states and Gaussians. Figure 5.1 shows the comparison of the F1 score using various state sizes and each state is modeled with different number of Gaussian distributions. The models did not show a huge difference by changing the number of states and Gaussians. All the three classifiers show relatively good performance with 6 hidden states modeled with one Gaussian. Also we can see that using 2 sates with 3 Gaussians, or 3 states with 2 Gaussians we can get nearly the same result.

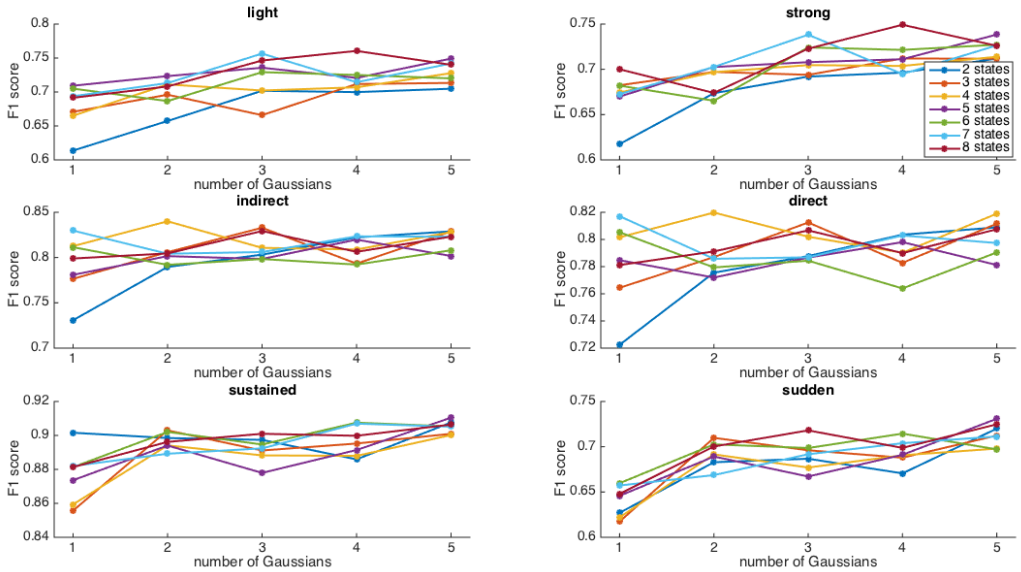

Figure 5.1: Comparison of the F1 score of classifiers using different number of states and Gaussians for each HMM

Regardless of Our experiments showed that using 8 hidden states and modeling them with 4 Gaussian mixtures resulted in the highest F1 score in identifying the weight of movement. We could also get nearly the same result using 7 states and 3 Gaussians. In the space classifier we could get the highest F1 score using 4 hidden states and 2 Gaussians. Finally the time classifier resulted in the highest F1 score using 3 hidden states and 2 Gaussians.

We also compared using ergodic and left-right architectures for the state

transitions of HMMs. The left-right architecture is described in section 1.2 of introduction. Figure 5.2 shows the comparison of the F1 scores of left-right model with various maximum state transition jumps, and the ergodic architecture. The experiment shows that the left-right architecture has better performance than the ergodic for the weight classifier regardless of the maximum allowable jump. However the left-right model with maximum allowable jump of one resulted in the lowest error rate comparing to the other left-right models. The space classifier is indifferent between the ergodic and left-right architecture with maximum jump of one and two. Also the time classifier is indifferent between the ergodic and left-right model with maximum jump of one.

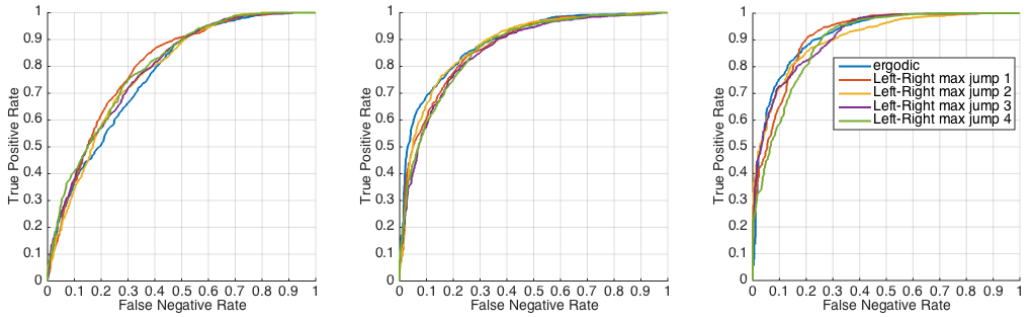

Figure 5.2: Comparison of the left-right state model with various maximum jump allowance and the ergodic model - positive class is indulging, negative class is fighting

We can conclude that the left-right architecture with maximum allowable jump of one is overlay a better choice comparing to the ergodic. This is because the left-right architecture has better performance for the weight classifier, and although it resulted in models with similar performance for space and time, the amount computation needed for an HMM with left-right architecture is less than ergodic. By using a left-right model with maximum jump of one, we can reduce the computation time for evaluating the likelihood of a sequence from N^2T to NT.

5.2 Failure Analysis

In section 4.2 of chapter 4 we discussed the windowing technique used for real-time gesture recognition. The Comparison of using different window sizes

for classifying time in figure 4.4 shows that the F1 score for classifying sudden gestures is decreasing with window sizes greater than 28. This is because the average length of the sudden gestures is 35 as we can see in table 3.1 and by increasing the window size further than 28 we already use the entire length of most of the gestures for train and test, and increasing the window size will not improve the sudden model. On the other hand the larger window size we use, the less samples we will have for testing. Therefore this will lower the accuracy of model because the fewer samples for testing will just increase the effect incorrectly classified sequences. Figure 5.3 compares the likelihood of sudden and sustained model on sudden gestures. As you can see the model does not improve using window sizes greater than 28, and also by increasing the window size we have fewer samples for testing.

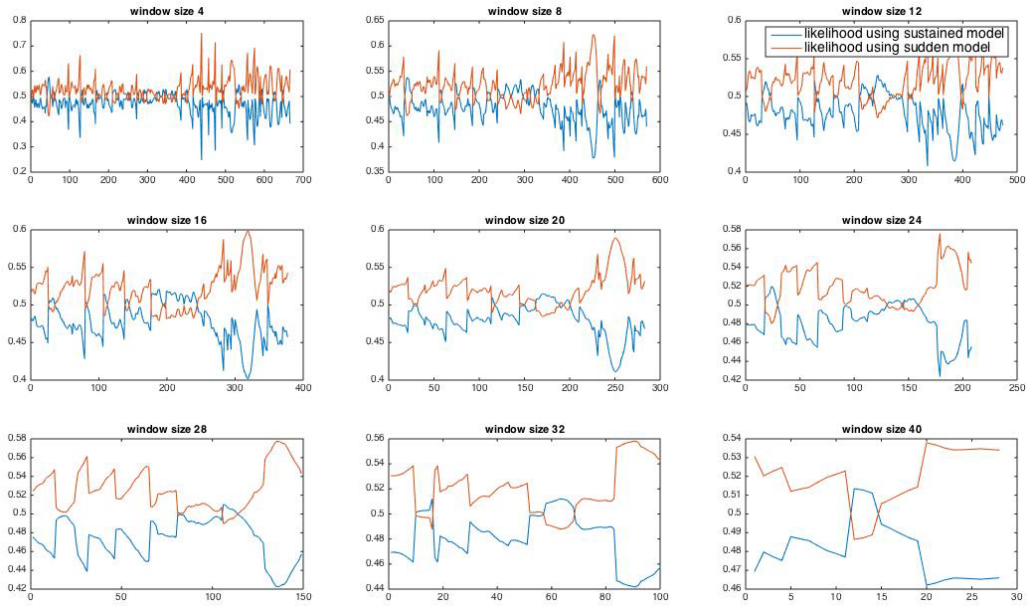

Figure 5.3: Comparison of the likelihood of the sudden gestures using the sustained and sudden HMMs with different window sizes

Another problem that we notice with the sudden gestures is their overall lower F1 score. The time classifier resulted in precision of 0.58 and recall rate of 0.93 for the sudden gestures. The high recall score means that the classifier can identify most of the sudden gestures, while the low precision means that the classifier classifies some parts of the gestures with sustained time as sudden. Figure 5.4 shows the speed of sustained gestures and the

likelihood of each HMM at that part. We can see that the likelihood of the errors are happened at the points where we have a sudden change in the speed of a sustained gesture as you can see with red marks. However this error is not too much to result in a low precision for the sudden classifier. By looking at the confusion matrix of the time classifier in table 5.1 we can see that this low precision is because the class of gestures with sudden time has less samples comparing the sustained class. This is because the sudden gestures are shorter than the sudden gestures.

		predicted	
		sustained	**sudden**
actual	**sustained**	1473	97
	sudden	10	139

Table 5.1: Confusion matrix of the time classifier

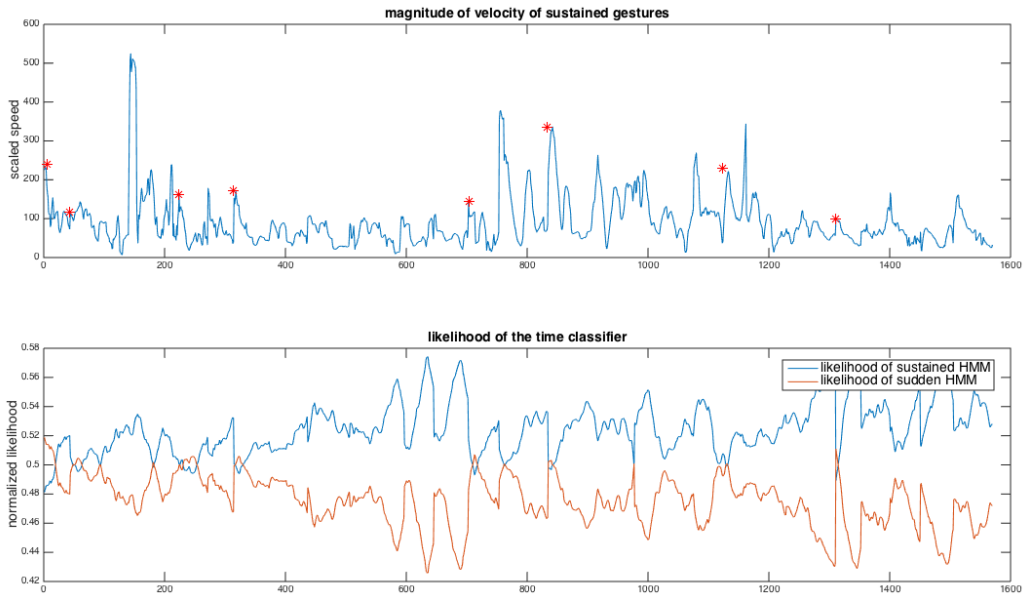

Figure 5.4: Likelihood of the time classifier on gestures with sustained time

In order to examine the effect of gesture length on the weight and space classifiers, we separate the sustained and sudden gestures and train the weight and space classifiers on the gestures of each group. Figure 5.5 shows the ROC curves of the classifiers that are trained and tested on gestures with similar time effort. The experiment resulted in better performance of the weight and

space classifiers. The performance is especially better when the models are evaluated on the gestures with sudden time which shows that the weight and space efforts are more distinct in sudden gestures comparing to sustained gestures. The experiment suggests that we might have better recognition of the BEFs if we first distinguish the time effort of gestures, and use that to choose the right model for classifying weight and space. However this also requires to have a good time classifier.

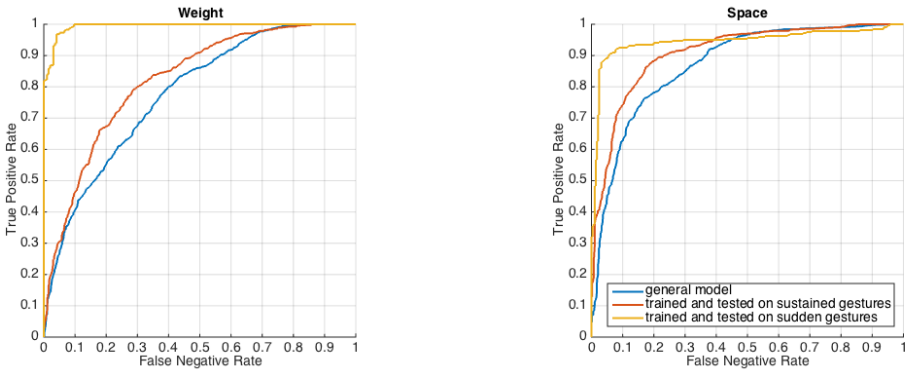

Figure 5.5: Comparison of the weight and space classifiers when evaluated on gestures with similar time effort and without knowing the time label - positive class is indulging quality, negative class is fighting quality

In another experiment we analyzed our movement data to identify the parts of gestures that have higher classification error. In this test we separate the gesture into 10 equal chunks and compute the classification error at each chunk. Figure 5.6 shows the comparison of error rate in identifying the BEFs using a window size of 32.

The analysis shows large error rates in the beginning and end of our gestures. By looking at the average norm of velocity of the gestures in figure 5.7, we realized an increase in this value at the end of most of the gestures. This problem is happened because of the way motion was captured in the recording sessions. In our recording sessions the LMA experts performed each BEA starting from the body center towards the top right and left corners, and retracted their hand with a constant speed regardless of the type of the gesture to the origin position. We eliminated the retractions from the gesture sequences and repeated the test. This could improve the performance as you

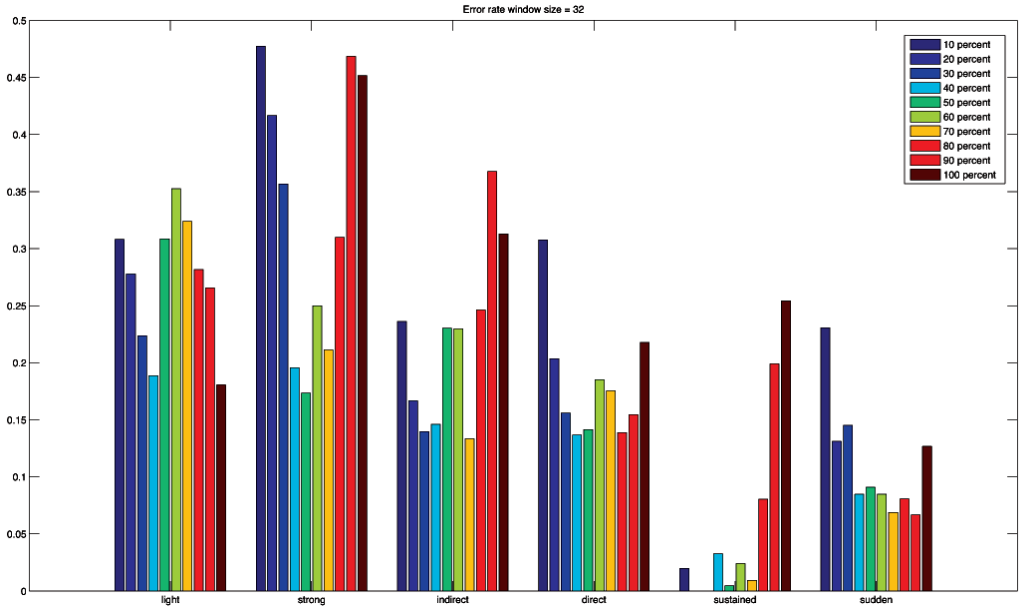

Figure 5.6: Error rate in each 10 percent chunk of the gestures

can see in figure 5.8.

5.3 Cross Dataset Evaluation

We evaluated the performance of the BEF classifier on the samples in KC dataset. The motion capture process for this dataset is described in chapter 3. In one experiment we trained the classifier on all the samples in SH dataset and tested on all the samples in KC dataset. This test resulted in very poor classification as you can see in figure 5.9. In another experiment we trained and tested the classifiers on a mixture of both datasets. This could improve the classification comparing to the previous experiment. However the F1 score is still lower than the model that was evaluated just on the SH dataset. We also evaluated the classifier just on the KC dataset. This test also resulted in lower F1 score comparing to the SH dataset. We can see that only the time classifier worked good on the both datasets.

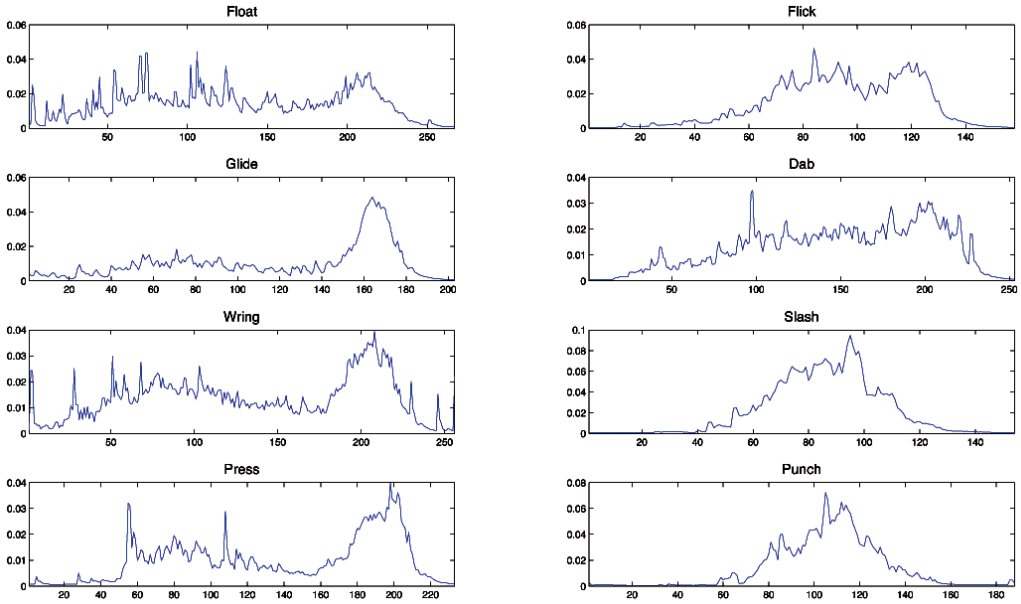

Figure 5.7: Average norm of velocity of the 8 BEAs

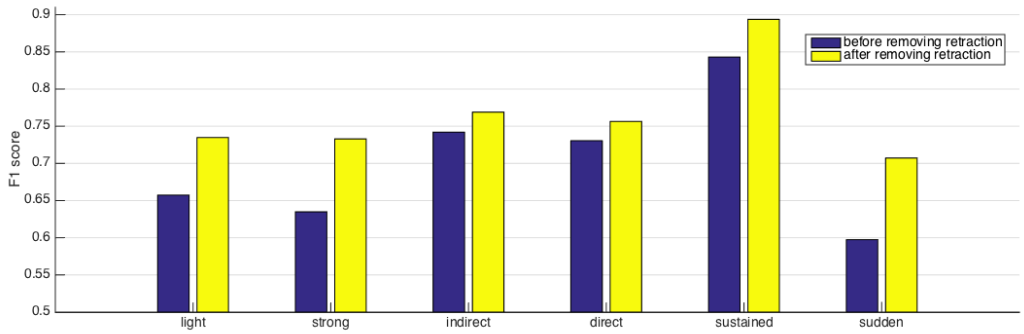

Figure 5.8: Comparison of F1 score after removing the retraction part from the gestures

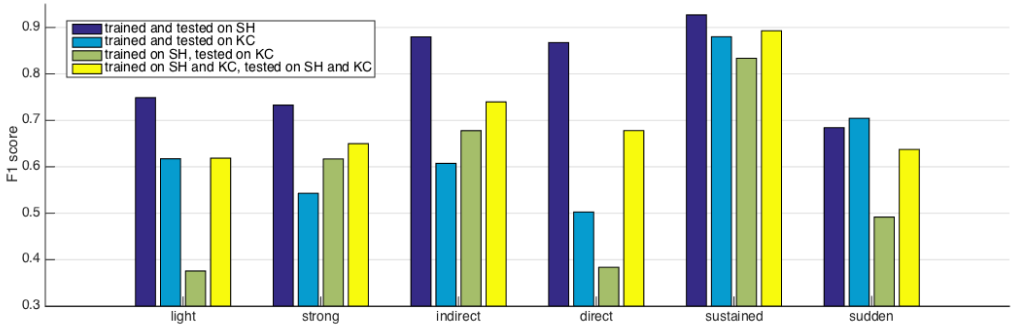

Figure 5.9: Evaluating the classifier on the KC dataset

CHAPTER 6

CONCLUSION

In this research we worked on modeling the expressions of hand movement using the Laban Movement Analysis (LMA) Framework. LMA describes movement with 4 Basic Effort Factors (BEFs) which are weight, space, time, and flow. Each BEF can have a continuous quality between two Indulging and fighting extremes. LMA defines 8 Basic Effort Actions (BEAs) where each action is made from an extreme state of the first three BEFs. The detailed description of LMA can be found in the introduction section. In this work we used Hidden Markov Models (HMMs) to model the BEFs of hand movement from various samples of the 8 basic effort actions.

We asked two LMA experts to perform the 8 BEAs facing toward the kinect camera with hands starting from the center of the body and reaching toward the right or left corner. Various motion derivatives, curvature features, and transformations are computed from the positional data of the hand joint. We identified salient features for each BEF and modeled them using a Hidden Markov Models (HMMs) for each indulging or fighting quality of the BEF. We compared the likelihood of HMMs to identify the most relevant quality of each BEF. In order to test the performance of model in real-time, we used a sliding window approach and segmented the gestures into smaller parts. We identified the window sizes that worked best for recognizing each BEF. We also compared different settings for the HMMs such as identifying the ideal number of hidden states Gaussian components, and comparing HMMs with left-right and ergodic state architectures.

Identifying expressive movement has an important part in developing systems that need a precise and natural understanding of the human movement, such as systems for recognizing musical gestures or in automated generation of computer animation. In this research we could successfully recognize the

time effort with 85 to 90 percent accuracy. The model also resulted in about 70 to 75 percent accuracy in recognizing the correct weight and space efforts. This research is still in the early stages of development. We have developed various tools and interfaces for extracting motion features, performing gesture recognition, analysis and comparing the models. In future we need to address various problems we faced in this work, such as identifying better features for the space and weight classifiers and developing methods for recognizing non-gesture parts in a movement sequence.

APPENDIX A

CURVATURE FEATURES

To compute the curvature features at each frame we first define **p1** and **p2** vectors in equation A.1a. The vectors show the position change prior to and after each frame. These vectors are also illustrated in figure .

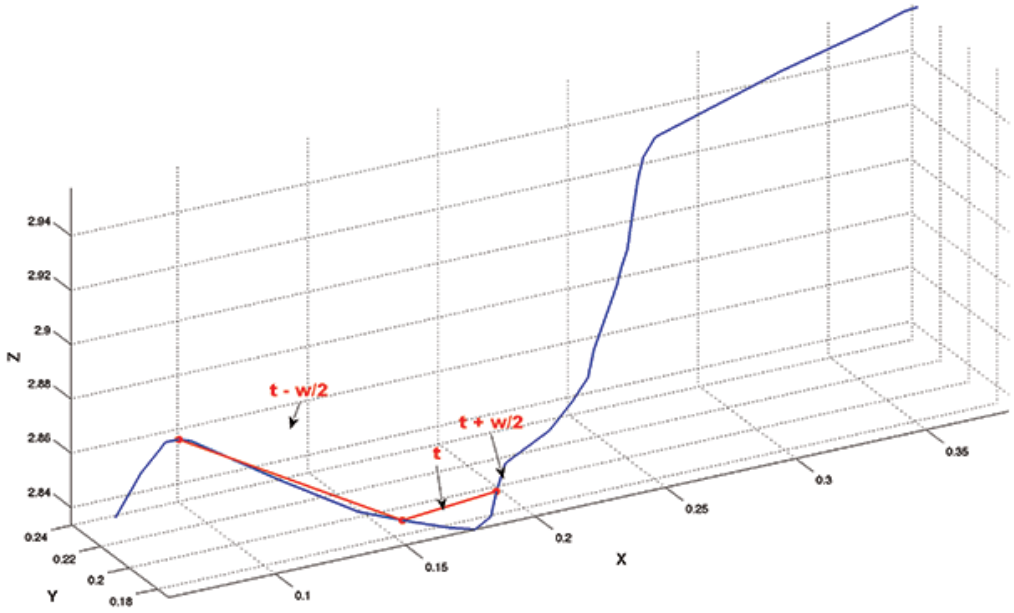

Figure A.1: By centering a window of size w we compute the dot product value between the vectors that connect the position at the center frame to the position at the beginning and end frames of the window

The cross product of **p1** and **p2** is a vector perpendicular to them and its direction follows the right hand rule. This vector can be computed using the determinant in equation A.1b. The magnitude of cross product is related to the length of the two vectors and the angle between them as you can see in equation A.1c.

Dot product of two vectors is a scaler value that is related to the length of the two vectors and cosine of the angle between them. In equation A.1d you can see the algebraic and geometric definition of dot product. This value is similar to the cross magnitude with exception that in cross magnitude we compute the $\sin\theta_t$ while in dot product we compute $\cos\theta_t$. Thus we will not use it directly as a feature and just use it to compute the angle between **p1** and **p2** in equation A.1e. The reason for using this equation instead of computing the angle directly from the cross magnitude is avoiding numerical accuracy around the values close to zero.

Another feature that is used to measure the curvature is the deviation of curve from a straight line defined in equation A.1f. With this measure we compute the difference between the L2-Norm and L1-Norm of the traveled distance along the temporal window, which can show the amount of deviation from a straight line during a specific period.

$$\begin{aligned} \mathbf{p1} &= \mathbf{p}_t - \mathbf{p}_{t-\frac{w}{2}} \\ \mathbf{p2} &= \mathbf{p}_{t+\frac{w}{2}} - \mathbf{p}_t \end{aligned} \tag{A.1a}$$

$$\mathbf{c}_t = \mathbf{p1} \times \mathbf{p2} = \begin{vmatrix} i & j & k \\ \mathbf{p1}^x & \mathbf{p1}^y & \mathbf{p1}^z \\ \mathbf{p2}^x & \mathbf{p2}^y & \mathbf{p2}^z \end{vmatrix} \tag{A.1b}$$

$$\|C_t\| = \|\mathbf{p1}_t\|\|\mathbf{p2}_t\| \sin\theta_t \tag{A.1c}$$

$$D_t = \mathbf{p1}_t \cdot \mathbf{p2}_t = \sum_{i=x,y,z} \mathbf{p1}_t^i \mathbf{p2}_t^i = \|\mathbf{p1}_t\|\|\mathbf{p2}_t\| \cos\theta_t \tag{A.1d}$$

$$\theta_t = \arctan\left(\frac{\sin(\theta_t)}{\cos(\theta_t)}\right) = \arctan(\frac{\|C_t\|}{D_t}) \tag{A.1e}$$

$$L_t = \sum_{\tau=t-\frac{w}{2}}^{t+\frac{w}{2}-1} \|\mathbf{p}_{\tau+1} - \mathbf{p}_\tau\| - \left\|\mathbf{p}_{t+\frac{w}{2}} - \mathbf{p}_{t-\frac{w}{2}}\right\| \tag{A.1f}$$

REFERENCES

[Alaoui et al., 2014] Alaoui, S. F., Françoise, J., Bevilacqua, F., and Schiphorst, T. (2014). Multimodal capture and representation of Laban Effort qualities. *ACM Trans. Comput.-Hum. Interact.*, 21(1).

[Bailey et al., 1994] Bailey, T. L., Elkan, C., et al. (1994). Fitting a mixture model by expectation maximization to discover motifs in bipolymers.

[Bevilacqua et al., 2010] Bevilacqua, F., Zamborlin, B., Sypniewski, A., Schnell, N., Guédy, F., and Rasamimanana, N. (2010). Continuous realtime gesture following and recognition. In *Gesture in embodied communication and human-computer interaction*, pages 73–84. Springer.

[Black and Jepson, 1998] Black, M. J. and Jepson, A. D. (1998). A probabilistic framework for matching temporal trajectories: Condensation-based recognition of gestures and expressions. In *Computer Vision—ECCV'98*, pages 909–924. Springer.

[Bouchard and Badler, 2007] Bouchard, D. and Badler, N. (2007). Semantic segmentation of motion capture using laban movement analysis. In *Intelligent Virtual Agents*, pages 37–44. Springer.

[Brand, 1997] Brand, M. (1997). Coupled hidden markov models for modeling interacting processes.

[Brand et al., 1997] Brand, M., Oliver, N., and Pentland, A. (1997). Coupled hidden Markov models for complex action recognition. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 994–999. IEEE.

[Chi et al., 2000] Chi, D., Costa, M., Zhao, L., and Badler, N. (2000). The EMOTE model for effort and shape. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 173–182. ACM Press/Addison-Wesley Publishing Co.

[Fine et al., 1998] Fine, S., Singer, Y., and Tishby, N. (1998). The hierarchical hidden Markov model: Analysis and applications. *Machine learning*, 32(1):41–62.

[Ghahramani, 1998] Ghahramani, Z. (1998). Learning dynamic Bayesian networks. In *Adaptive processing of sequences and data structures*, pages 168–197. Springer.

[Laban, 1980] Laban, R. (1980). *The mastery of movement.* Plymouth: Macdonald & Evans.(Original work published 1950).

[Lee and Kim, 1999] Lee, H.-K. and Kim, J.-H. (1999). An hmm-based threshold model approach for gesture recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(10):961–973.

[Levine et al., 2010] Levine, S., Krähenbühl, P., Thrun, S., and Koltun, V. (2010). Gesture controllers. *ACM Transactions on Graphics (TOG)*, 29(4):124.

[Murphy, 2002] Murphy, K. P. (2002). *Dynamic Bayesian networks: representation, inference and learning.* PhD thesis, University of California.

[Rabiner, 1989] Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[Rett et al., 2010] Rett, J., Dias, J., and Ahuactzin, J.-M. (2010). Bayesian reasoning for laban movement analysis used in human-machine interaction. *International Journal of Reasoning-based Intelligent Systems*, 2(1):13–35.

[Siddiqi et al., 2007] Siddiqi, S. M., Gordon, G. J., and Moore, A. W. (2007). Fast state discovery for hmm model selection and learning. In *International Conference on Artificial Intelligence and Statistics*, pages 492–499.

[Smith et al., 1997] Smith, S. W. et al. (1997). *The scientist and engineer's guide to digital signal processing.* California Technical Pub. San Diego.

[Vasilescu, 2002] Vasilescu, M. A. O. (2002). Human motion signatures: Analysis, synthesis, recognition. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 3, pages 456–460. IEEE.

[Zhang et al., 2004] Zhang, D., Gatica-Perez, D., Bengio, S., McCowan, I., and Lathoud, G. (2004). Modeling individual and group actions in meetings: a two-layer hmm framework. In *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on*, pages 117–117. IEEE.

[Zhao and Badler, 2005] Zhao, L. and Badler, N. I. (2005). Acquiring and validating motion qualities from live limb gestures. *Graphical Models*, 67(1):1–16.